浙江省大学生
人工智能竞赛
AI 赋能未来

中国移动杯·第一届
浙江省大学生人工智能竞赛

创意赛报告

☑ 钉钉专项赛
☐ 网易专项赛
☐ MaxKB 专项赛

队伍名

中国移动杯·第一届
浙江省大学生人工智能竞赛

钉钉专项赛

初赛报告

作品名称：钉钉 AI 日程规划精灵

摘要

在当今快节奏的社会中，人们常因繁杂的日程安排而感到困扰，难以高效规划时间并平衡工作与生活。为此，团队开发了基于钉钉平台的智能助手——“日程规划精灵”，旨在通过人工智能技术解决日程管理中的核心痛点，提升用户的时间利用效率与生活品质。

该应用**具备多项实用功能**：**一是智能问候与引导**，通过欢迎词与选项帮助用户快速了解并使用核心功能；**二是连续创建新日程，支持多日程一站式创建**，克服了平台原生功能单一、无法批量处理的局限；**三是空闲时间推荐**，结合用户已有日程与大模型分析，智能推荐无冲突的时间段供用户选择；**四是通过定时触发与提醒**，防止用户遗忘重要事项；**五是基于大模型的社交辅助功能**，可自动生成生日祝福、活动文案等，减轻用户在社交中的精力投入；**六是日程冲突智能调整**，通过分析日程优先级，协助用户重新安排冲突任务。

在技术创新方面，团队提出了**“AI 决策 + 代码规范化变量 + 模块执行”**的融合开发思想。该模式首先利用大模型（如 Deepseek）从自然语言中提取关键信息并转化为结构化变量，再通过代码对 AI 输出的 JSON 文本进行校验与规范化处理，最终由平台模块实现功能执行。这一流程既保留了大模型对自然语言的理解优势，又通过代码保障了程序的严谨性与可控性，显著提升了系统的可靠性与灵活性。

此外，团队还提出了**“宏观循环与微观公式代码多维嵌入”**的开发思路。在宏观层面，通过嵌套循环（如课程数量 × 周数）实现大批量日程的自动生成；在微观层面，通过在模块中嵌入轻量级代码或公式，实现变量的高效运算与控制。这两种方法不仅解决了复杂场景下的批量处理难题，也降低了开发门槛，拓展了低代码平台的适用性。

总之，“日程规划精灵”不仅在功能上实现了智能化、一体化的日程管理，更在技术路径上探索出融合 **AI 与代码控制**的开发范式，具备良好的实用性与推广价值，为智能体开发提供了可借鉴的经验。(如图 1 所示)。

关键词：连续创建新日程，定时触发与提醒，代码规范化变量，多维嵌入，AI 与代码控制

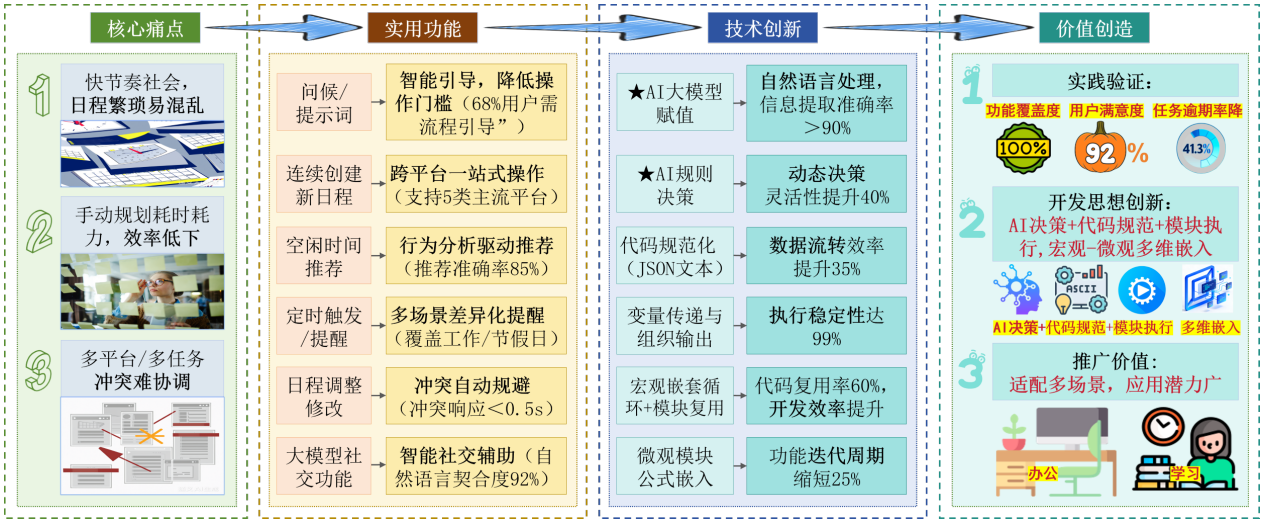


图 1：“日程规划精灵”总设计思路

目录

一 “日程规划精灵”的应用背景

当今社会，快节奏的数字浪潮推动生活与工作节奏持续加快，繁杂的工作会议、学习安排，以及同学聚会、社团活动等各类事务，常让人们陷入焦头烂额的困境——既难以从众多事项里高效抉择，又不知如何对必须完成的事务做合理且高效的日程安排，更渴望在完成本职任务的同时，能拥有闲暇时光与亲朋友好友相伴。

针对这一时代痛点，“钉钉 AI 日程规划精灵”直击**需求核心**，从时间规划与日程安排的角度切入，为人们带来技术赋能的便利。出于对这类生活困扰的关注，团队设计了这款人工智能钉钉助理——“钉钉 AI 日程规划精灵”，致力于帮助学生与各行业从业者解决面对复杂日程时手忙脚乱的问题，简化日程安排中的繁杂流程。

二 “日程规划精灵”的实用功能

实质上，“日程规划精灵”是一个融入了大模型与代码等基本控制单元的智能程序，包含了完整的控制流；**输入量即为用户的对话自然语言文本信息**，再通过智能助手的语义分析决策，**选择不同的备选工作流**，在工作流中进行自然语言的解析、变量的定义与控制（包括日程创建变量）及对话返回内容的组织，同时以对话自定义记忆、推理增强等功能进行加强辅助，最后以**返回对话文本、日程创建等作为输出量**，由此实现完备的程序控制。

在此基础之上，通过分析测试平台自带工作流不足之处（包括多日程无法一站式创建等）、挖掘日程规划的潜在需求（包括空闲时间推荐、日程速览导出等），进而实现智能体开发的实用功能创新。以下文段将对智能助手的各个实用功能进行展开介绍。

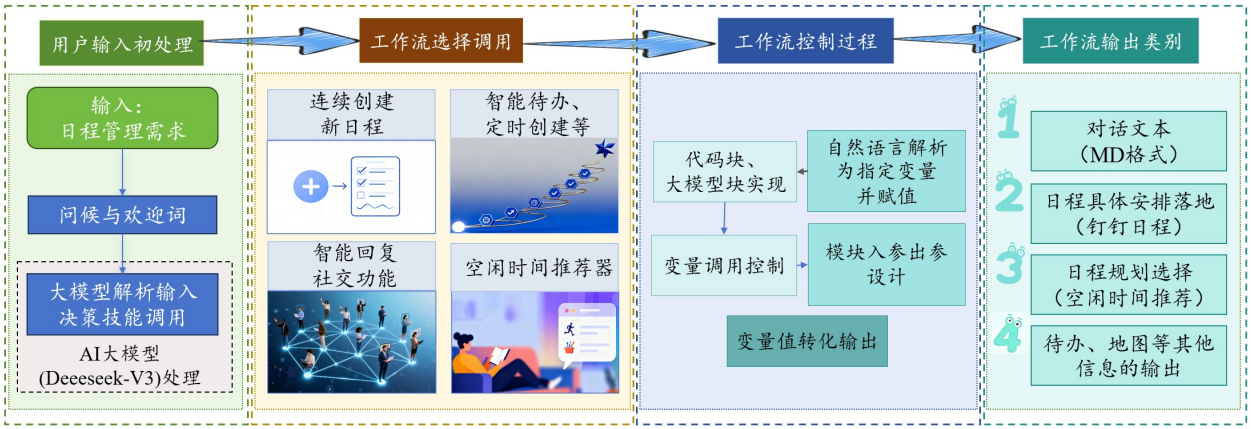


图 2: “日程规划精灵”程序控制流及功能概览

2.1 问候和提示词

为了让用户了解该智能助手的基本功能，在“日程规划精灵”启动并进入对话页面时，团队**设计了欢迎词**以介绍智能助手的基本功能以及用法的信息，并在下方以选项的形式提供初次使用的用户去体验其基本功能，包括了日程规划实例，社交功能体验，突发日程调

整演示以及自动规划当天行程。用户通过点击对应选项，智能助手会以**文本形式**进行初步的设计与回答，以此让用户体验并了解助手的实用性。

(a) 问候语和欢迎词　　**(b)** 日程规划实例

图 3: “日程规划精灵”功能展示

2.2 连续创建新日程

在开发过程中，为了充分运用并理解平台开发原生功能，团队对其自带的日程相关模块**进行了系统性测试**，发现了无论是对话内嵌的助手，还是预设的“新建日程”与“更新日程”工作流，均在交互性、功能完备性和逻辑准确性上存在明显不足，**平台自带功能缺陷分析**如下所示。

表 1: 钉钉平台自带日程相关功能缺陷概述

功能模块	主要缺陷
对话内嵌助手（日历/待办）	无法响应对话指令，需用户手动添加，交互性差。
“新建日程”工作流	功能过于精简，仅支持单一日程创建，且不支持持续时间、重复规则（如每月重复）、用户记忆的优先级标签（如工作 > 个人 > 娱乐）、冲突检测等功能。
“更新日程”工作流	无法连续创建多个日程，确认流程繁琐，存在日期语义解析的逻辑错误。

(a) 日历助手的对话响应　　**(b)** 待办助手的对话响应

图 4: 钉钉已有对话中，助手对于对话输入的无响应

(a) 平台自带的“日程助手”工作流的对话响应　　**(b)** “日程助手”创建日程的响应

图 5: 钉钉智能体平台中，已有工作流实现创建日程的功能

图 6: 平台自带“新建日程”工作流，对于多日程请求仅响应一个

图 7: 平台自带“更新日程”工作流的逻辑错误（10 月 1 日的明天仍是 10 月 1 日）

本团队设计的“创建新日程”工作流，针对以上问题逐一进行了优化，面对处理多日程问题，在此加持之下，“日程规划精灵”明显优于其他一般的助手；为节省篇幅，以下**择要展示一连串办事流程的一站式日程创建**，如下图所示。

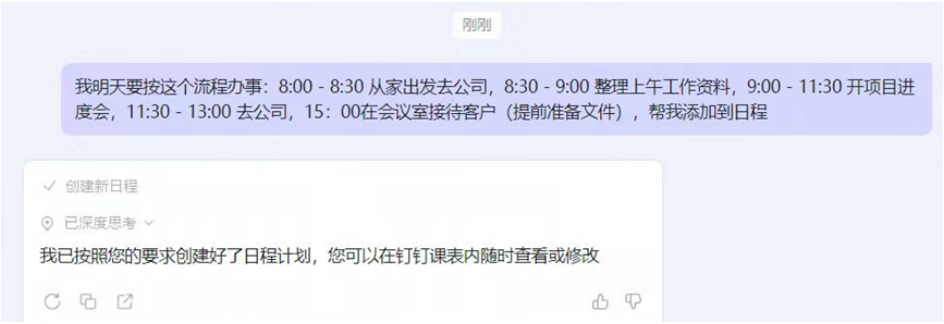


图 8: 团队设计“创建新日程”工作流的对话响应

图 9: 团队设计“创建新日程”工作流，成功创建了一连串日程；对比上述自带工作流，功能可靠性大大提升

2.3 空闲时间推荐

团队通过对日程规划的实际需求的思考挖掘，发现在日常生活中，人们常常会出现对于一个将要去做的事物（比如学习数学），**只有模糊的打算**（如计划在明天下午学习），有时**也不清楚是否存在日程冲突**；针对这一需求，团队设计了“空闲时间推荐器 · AI 智能决策”工作流，通过对用户已有日程的导入分析，结合**大模型的智能决策，推荐出两项不存在日程冲突且符合用户需求的日程安排，并由用户最终选择规划方案**。具体工作流成效如下图。

图 10: 团队设计“空闲时间推荐器”工作流的对话响应，设计按键选择方案

图 11: 团队设计“空闲时间推荐器”工作流的日程安排结果，规避了日程冲突

2.4 定时触发与提醒

智能助手还新增了定时触发功能，用户可以通过与 AI 助理的对话开启、查询以及关闭定时触发任务；**定时触发任务也会在特定时间提醒用户临近的日程信息**，以此避免用户遗忘。

图 12: 智能助手的定时触发功能

2.5 基于大模型的社交功能

智能助手通过调用平台自带的智能回复的自然语言功能，用户可以**发送各类社交的请求**（如生日祝福构思、社团破冰游戏设计等），以此分担用户在社交过程中的精力投入。

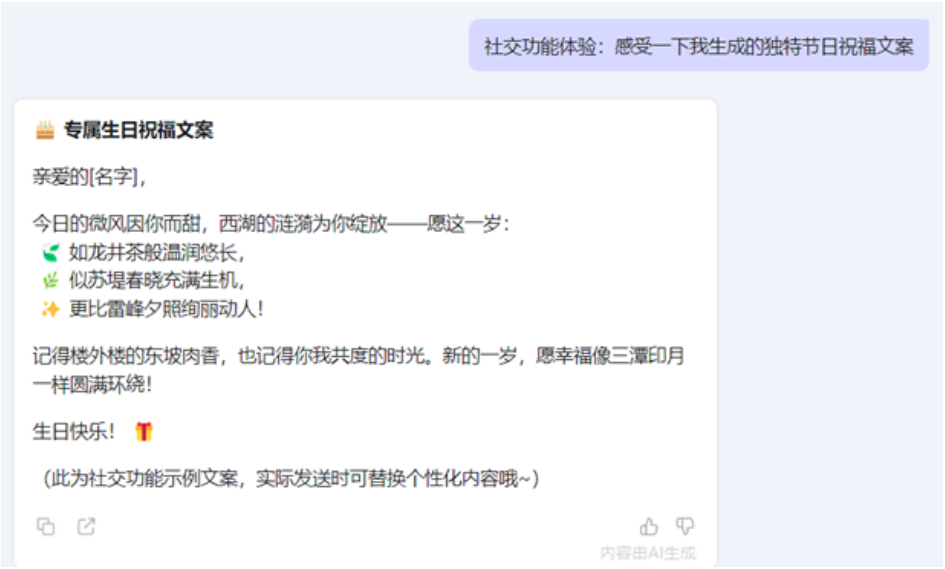


图 13: 智能助手的社交功能（以生日祝福为例）

2.6 日程调整修改功能

当日程出现冲突时，用户可以向智能助理提供冲突的日程以及时间，AI 助理会通过**分析日程重要程度做出取舍**或者其他时间的调整。

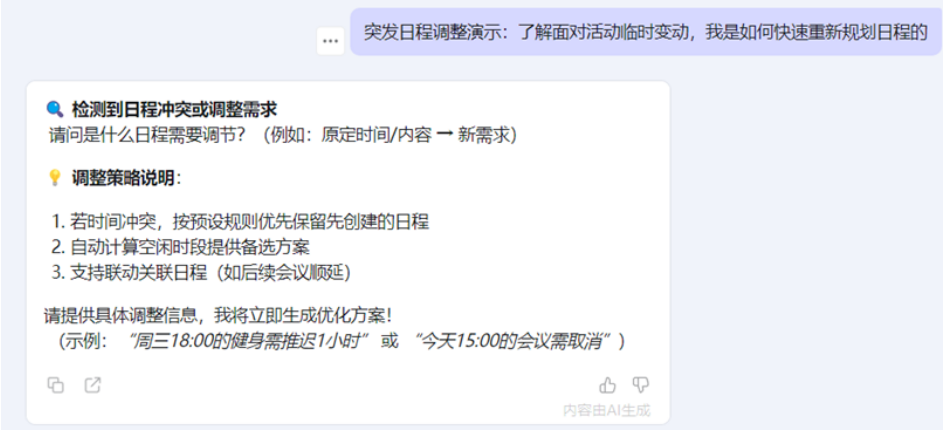


图 14: 智能助手的日程调整修改提示

三 “日程规划精灵”的技术创新

上述段落介绍了“日程规划精灵”的实用功能，**以何满足用户复杂多变的各类需求已成为信息产业的行业难题，传统的非大模型的代码控制与单一的大模型文本决策将无法保证程序的可靠性要求**。团队汲取传统方式的优点，研究并克服了老办法的缺陷，融合团队成员对钉钉智能体平台的熟练开发的技能与认识，提出了新的智能体技术设计思维，包括了 AI 决策结合代码后处理的控制流，以此深入挖掘了智能体开发的应用价值与潜力，以下文段将具体介绍本团队**原创的新开发思想**。

3.1 “AI 决策 + 代码规范化变量 + 模块执行”工作流融合开发思想

“日程规划精灵”的用户输入信息，最关键的就是其**自然语言的对话指令**，如“明天下午我要在家学习数学，请帮我推荐空闲时间”，对于此类指令，由于语言的多样性，其**输入并非结构严格**，传统 python 等代码对于此处理将十分困难，后续的日程决策更是无从谈起。因此对于这类自然语言输入，在工作流的设计中，团队采用了 Deepseek-671B 大模型，首先利用钉钉平台的“调用工作流”模块，通过变量的预先设置（包括名称、类型与变量说明），使得大模型可以**提取自然语言输入的有效信息，赋值给这些变量**，为后续的决策做准备。

(a) 工作流中“调用工作流”模块的变量预设　　**(b)** 利用调试界面，“调用工作流”模块确实成功提取变量值

图 15: 利用大语言模型，从自然语言中提取预设变量信息

随后，在获得结构化变量信息后，由于用户要求的灵活多变性，**其规则难以用代码进行约束，日程规划依旧需要 AI 决策**。本段以“空闲时间推荐器”工作流为例，在 AI 决策前还需要利用“查询日程列表”读取用户已有规划日程，至此决策输入即为已规划日程与自然语言结构化变量，在结合 AI 决策的具体要求、约束、待输出 JSON 格式等预设 prompt，以下是 prompt 重点片段，此决定了 AI 决策的功能，依托 AI 决策即可替代传统低效的纯代码决策。以下是 prompt 的部分展示与 AI 决策结果示例，通过 prompt 设置，即可成功地进行决策。

Listing 1: “空闲时间推荐器”工作流中 AI 决策的部分 Prompt

```
你是一个日程检查助手。
现在给你一个候选的空闲时间推荐结果，请严格生成JSON的推荐结果，并检查是否符合用户的原始偏好要求。
任务标题：task_title；时长（分钟）：duration_min；
安排范围：date_scope；偏好时段：prefer_windows；截止时间：deadline；候选数candidate_count；地点偏好：
    location_pref
候选推荐结果（JSON），需要你来生成
{
  "task_title": "string，任务标题",
  "duration_min": 120,
  "date_scope": "string，对应原始安排范围（如：明天 / 2025-10-01 ~ 2025-10-03）",
  "prefer_windows": ["13:30-17:30"],
  "deadline": "string，可为空，例如 2025-10-03 23:59",
  "location_pref": "string，可为空",
  "candidate_step_min": 30,
  "candidates": [
    {
      "start": "2025-10-01 13:30",
      "end": "2025-10-01 15:30",
      "note": "符合偏好时段"
    }
```

```
 ],
 "summary": "string, 供展示的摘要文本"
}
输出最终合法的候选时间 JSON, 不要解释额外文字。
若用户指定了“上午/下午/晚上/早上/凌晨”等, 则候选时间必须严格落在对应时间段内: 上午 = 08:00 - 12:00;
下午 = 13:30 - 17:30; 晚上 = 18:00 - 22:30; 早上 = 08:00 - 10:00; 凌晨 = 00:00 - 06:00; 若不满足, 剔除该
    候选时间。
1. 冲突与冗余
忙碌冲突: 候选与返回内容 (已存在日程) 不能重叠。重叠时直接剔除。
候选重复: 对于完全相同或高度重叠 (例如起始时间相差 < 5 分钟) 的候选, 可只保留一个, 避免卡片里看起来一
    样。
候选数量上限: 输出候选数不能超过 candidate_count; 若不足, 可以在 summary 里说明“只找到 1 个满足条件的
    时间”。
......(其他约束略)
```

Listing 2: AI 决策的结果示例（用户输入：“明天下午我要在家学习数学，请帮我推荐空闲时间”）

```
{ "task_title": "学习数学", "duration_min": 60, "date_scope": "明天", "prefer_windows":
    ["12:00-18:00"], "deadline": "", "location_pref": "家", "candidate_step_min": 30, "candidates
    ": [ { "start": "2025-10-02 13:00", "end": "2025-10-02 14:00", "note": "符合偏好时段" }, { "
    start": "2025-10-02 15:00", "end": "2025-10-02 16:00", "note": "符合偏好时段" } ], "summary": "
    找到2个符合条件的时间" }
```

在得到 AI 决策输出的有效 JSON 文本后，由于这些决策没有形成工作流内的结构化变量，其在工作流中还不能直接利用，因此**需要结合 python 代码，将 JSON 格式的文本输出进行转译**，同时再一次检查 AI 输出的 JSON 是否符合基本的逻辑，以免出现如“10 月 1 日的明天是 10 月 1 日”的钉钉自带工作流的错误。以下是代码核心控制函数，以此保证 JSON 转化的有效性。

Listing 3: “空闲时间推荐器”中，清洗 AI 决策 JSON 文本转工作流可用 JSON 的核心函数

```
def validate_schedule(schedule_json: str) -> Dict[str, Any]:
    """
␣␣␣␣主校验函数: 解析JSON, 并根据约束条件筛选有效的候选日程。
␣␣␣␣Args:schedule_json:␣AI模型返回的包含日程规划的JSON字符串。
␣␣␣␣Returns:一个字典, 包含有效的和无效的候选日程列表及总结。
␣␣␣␣"""
    try:
        data = json.loads(schedule_json)
    except json.JSONDecodeError:
        raise ValueError("输入不是有效的JSON格式")
    # 1. 提取约束条件
    duration_min = data.get("duration_min", 0)
    prefer_windows = data.get("prefer_windows", [])
    prefer_segments = _collect_prefer_segments(prefer_windows)
    # 2. 遍历并校验所有候选方案
    valid_candidates: List[Dict] = []
    invalid_candidates: List[Dict] = []
    for cand in data.get("candidates", []):
```

```
        reasons = []
        start_dt = _parse_dt(cand["start"])
        end_dt = _parse_dt(cand["end"])
        # 2.1 校验时长
        actual_duration = (end_dt - start_dt).total_seconds() / 60
        if duration_min and actual_duration < duration_min:
            reasons.append(f"时长不足（要求{duration_min}分钟，实际{actual_duration:.0f}分钟）")
        # 2.2 校验偏好时段
        if not _is_within_segments(start_dt, end_dt, prefer_segments):
            reasons.append("不在偏好时段内")
        if reasons:
            invalid_candidates.append({"candidate": cand, "reasons": reasons})
        else:
            valid_candidates.append(cand)
    # 3. 组装并返回结果
    return {
        "task_title": data.get("task_title", ""),
        "constraints": {
            "duration_min": duration_min,
            "prefer_windows": prefer_segments,
        },
        "valid_candidates": valid_candidates,
        "invalid_candidates": invalid_candidates,
        "summary": f"有效方案␣{len(valid_candidates)}␣个，无效方案␣{len(invalid_candidates)}␣个。",
    }
```

通过代码的规范化变量格式，可以**从复杂与嵌套的 AI 决策 JSON 文本里提取出任务日程，并验证合法性**，再输出结构化汇总结果，通过此，即可提取出工作流中有效的变量值。

(a) 利用“执行代码”模块，成功提取“AI 决策”中的有效变量　　**(b)** 有效变量提取后，变量传递进入“消息通知”输出执行模块

图 16: 代码提取有效变量后，决策变量真正的落地执行

最后，**利用钉钉平台的“AI 助理回消息给当前用户”模块功能，充分利用前面的决策与赋值的变量，组织出回复的语句，并充分利用平台预置的按钮与创建钉钉日程功能**，“空闲时间推荐器”工作流即可成功完成智能时间决策功能，并给用户已多元化且合理的选择，如图 10的对话输出所示，其控制流程如下。

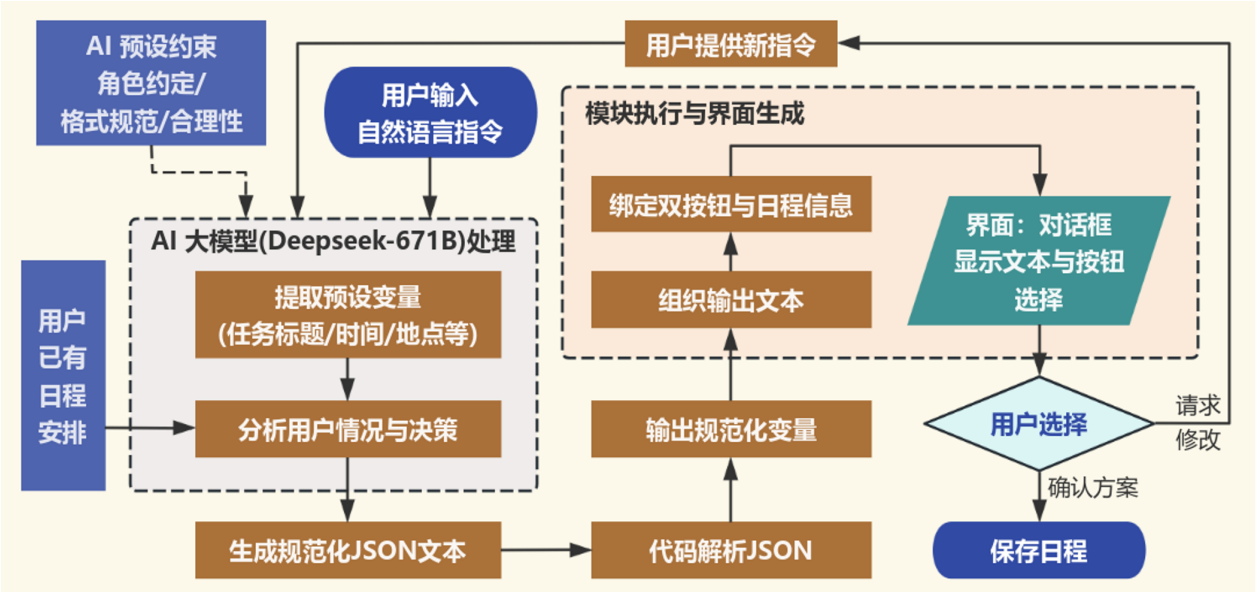


图 17: “AI 决策 + 代码规范化变量 + 模块执行”开发思想的控制流程（以“空闲时间推荐器”为例）

同时在本团队的**另一个“课表导入日程”工作流**，由于平台对于“循环内容”此模块输入的 JSON 格式要求，为了有效地组织起控制处理，团队在开发此工作流时再次应用此

“AI 决策 + 代码规范化变量 + 模块执行” 的开发思想，将 “调用工作流” 控制块中，**AI 对于用户输入的自然语言的提取与预置文本变量的赋值，利用代码模块对此进行 JSON 化处理，并将输出 JSON 的复用到后续的 “以科目数组循环” 等模块中**。此步骤在上述工作流中，奠定了所有的课表日程创建的控制基础，是典型的该开发思想的的**轻量级应用**。此开发思想通过两类不同功能的工作流对于的诠释与应用，发现的确能够在工作流中，组织起有效的变量传递，证明了本节所述的开发思想具有广泛的应用价值与推广潜力。

Listing 4: “课表导入日程” 工作流中，轻量化的 JSON 组织代码

```python
def main(params: dict):
    # 用于存储最终要返回的科目名称列表
    result_list = []
    # 循环从 name1 到 name10，依次获取入参里对应的值
    for i in range(1, 11):
        # 拼接当前要获取的变量名，比如第一次循环是 "name1"，第二次是 "name2" ，直到 "name10"
        param_name = f"name{i}"
        # 从 params 这个字典里获取对应变量名的值，params 里存着配置的入参数据
        value = params.get(param_name)
        # 如果获取到了对应的值（值不为空），就把它添加到结果列表里
        if value:
            result_list.append(value)
    # 将整理好的包含 10 个科目名称（如果有的话）的列表作为结果返回，这个列表在输出时会自动转成 JSON 格式的数组
    return result_list
```

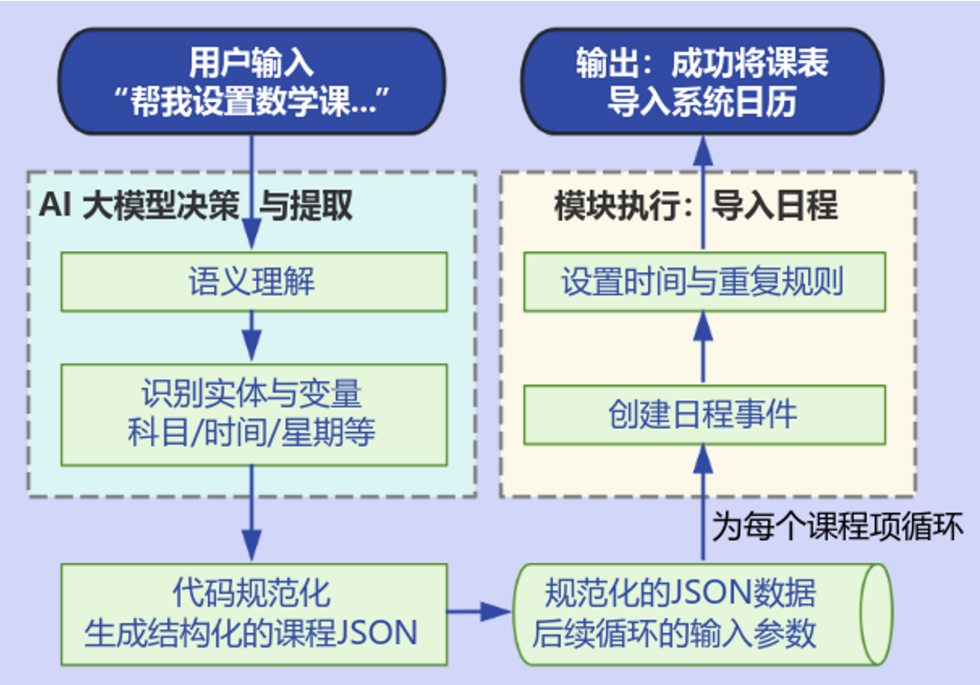


图 18: “AI 决策 + 代码规范化变量 + 模块执行” 轻量级思想应用（以 “课表导入日程” 部分控制为例）

相对于仅传统代码开发或仅 LLM 模型应用，此类开发思路下设计的程序**性能与稳定性得到极大的提升**；通过钉钉智能体平台将两者**融合开发**，此思路**汲取了大模型的对于自然语言的充分理解与代码的逻辑严谨与严密控制的各自优点**，利用**控制流与模块的设计**将两者充分融合，最终完美实现了日程决策的智能化功能。此思路因其合理性完全可以**大规模的推广**；在基于 COZE 平台的 ERP 商战沙盘系统等其他智能体应用与开发场景中，团队的开发人员亦用此进行智能助手的设计（先进行代码的历史决策与规则的 JSON 规范化，大模型设置 prompt 后，读取放在云端数据库的 JSON 向量进行新规则辅助决策）。

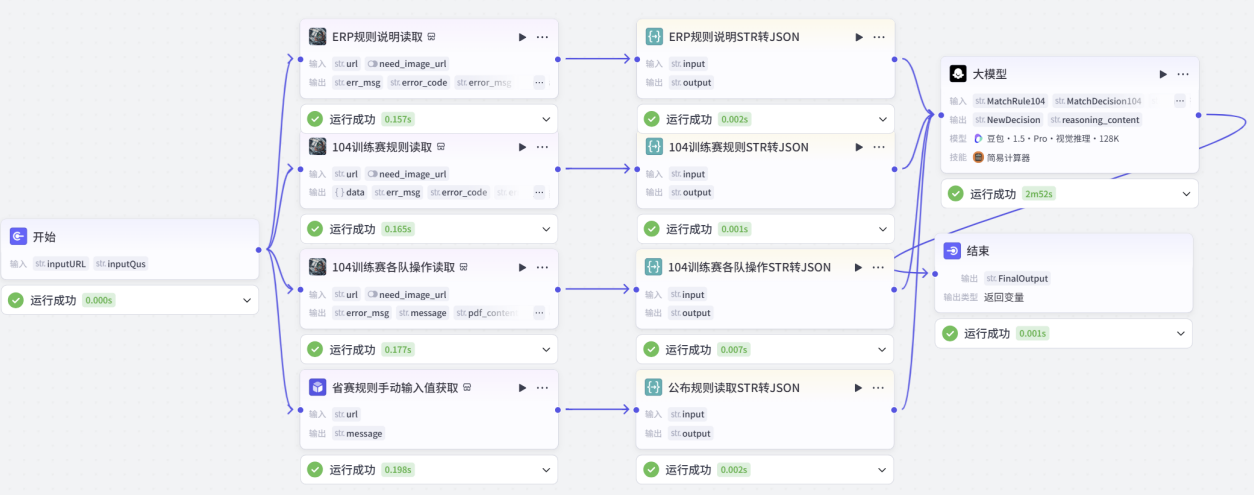


图 19: 开发思想的平台与应用领域推广：基于 COZE 平台的 ERP 商战历史数据学习与辅助决策，将应用于相关的**国家社科基金一般项目**

3.2 宏观循环与微观公式代码多维嵌入的工作流开发思想

复杂控制的建立与合理决策的输出并非只需要依托“AI 决策 + 代码规范化变量 + 模块执行”此类在控制执行与维护上的智能体融合开发思想的创新，还需要从**“嵌入”“嵌套”这一基本开发思想**出发，在宏观与微观上针对灵活多变的用户需求进行工作流设计的技术创新，以下文段将从**宏观循环**、**微观代码两种不同维度**的“嵌入”设计进行展开。

3.2.1 多层嵌套宏观循环

在实际的应用场景中，与传统的代码开发相似，**单一的顺序与分支控制难以满足所有的开发需求**，或者说即使满足了，程序的复用性将会非常的不理想，同时这也非常不适合大批量的控制对象。团队在开发“课表导入日程”工作流时发现，学生群体用户往往会在一周有大约 10 种不同科目的课程（假设有 n 种课程），同时这些课并非只上一次，一学期上大约 18 周左右（假设为 m 周），那么若要创建出完备的日程规划，足足需要 $n \times m$ 个全学期日程创建（约 180 个日程），这是一般的顺序控制的工作流无法实现的，**最关键的是顺序控制难以建立 n 与 m 数量变化的动态响应**。针对这一开发瓶颈，团队**创造性地提出了将多层嵌入宏观循环**作为“课表导入日程”工作流的宏观控制过程，概括来看就是**以课**

程数量 n 作为大循环，以课程周数 m 作为小循环，以此进行全学期日程的足量创建。以下是嵌套循环批量创建日程的控制流程。

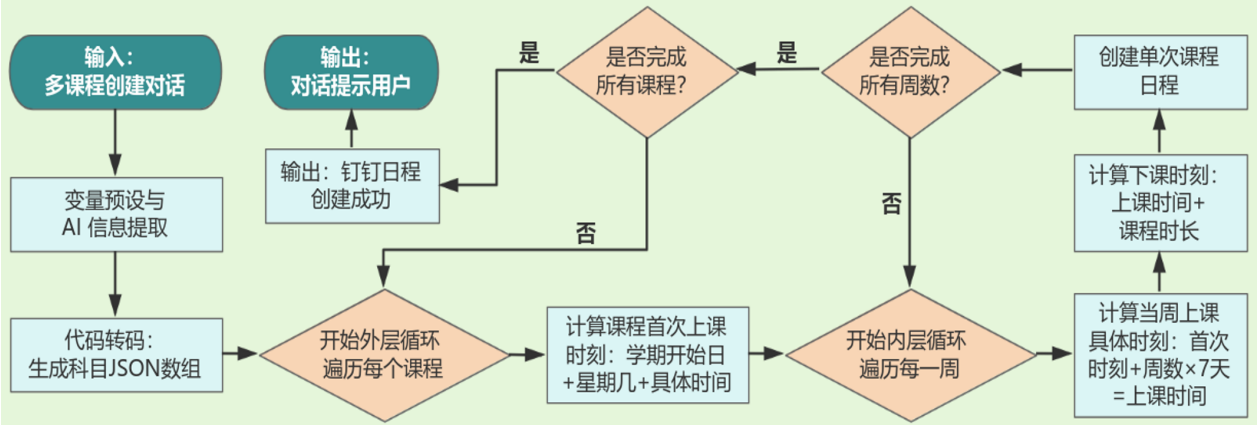


图 20: “课表导入日程”工作流的宏观循环嵌套的控制流程

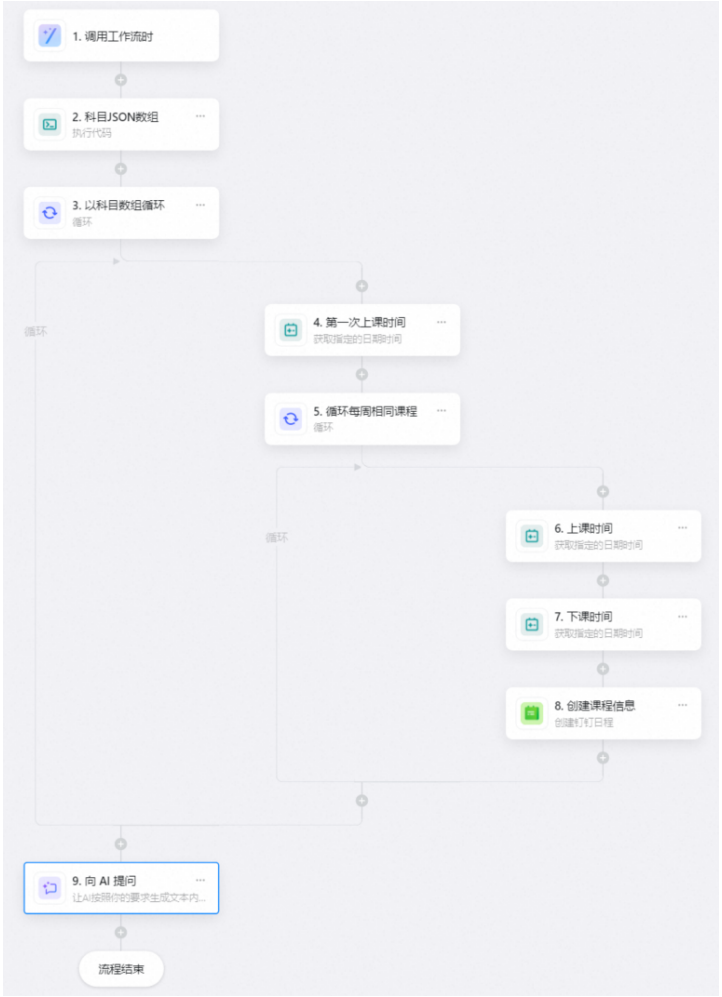


图 21: “课表导入日程”工作流的宏观循环嵌套，在钉钉智能体开发平台中的展现

通过这样的循环嵌套，建立了课表导入日程的基本控制框架，这也**对所有的智能体开发中，大批量的重复控制工作的复用具有一定的启示意义**。比如在企业销售终端管理中，可用建立针对销售门店外层循环、针对每日时段内层循环批量生成巡店提醒。

3.2.2 工作流模块内的微观公式代码嵌入

在 20中，最终获得的逐课程、逐周的日程信息涉及到输入已处理的日程信息（如当前课程的第一次上课时刻、课程的时长）、日程的计算（当前周当前课程的开始时间），这些显然需要**变量的处理运算**。针对此，团队充分利用平台的“获取指定的日期时间”模块的变量复用与代码/公式的嵌入功能；以外循环的当前课程“第一次上课时间”的计算为例，在提取了用户对话中的“学期开始时间”的时刻信息，再利用代码，比对本次 JSON 列表的当前课程循环，代码运算输出此课程的首次授课时刻相对于“学期开始时间”的时刻偏移量，即可计算出当前课程的“第一次上课时间”。**控制过程相当的简练，相对于上一章节的 AI 决策 JSON 转工作流标准 JSON 的上百行代码有了大量减少**；但本节关注的是，团队以此设计作为模块内的**微观公式代码**嵌入开发可行性的验证与探索，这种对**模块灵活设计的开发思想是完全可落地的**，其设计产品在经过调试、修改后就具有实用性；同时在开发难度上，由于**低代码平台的开发者水平层次不齐**，并不是所有开发者都能接受进 400 行的 python 代码开发难度，**此类用户可以在本节的“微观公式代码嵌入”到“微型公式代码嵌入”的思想指导下，亦可设计出高效的工作流**，这大大拓展了智能体平台的受众群体。以下对于此工作流的有效性，进行测试结果展示。

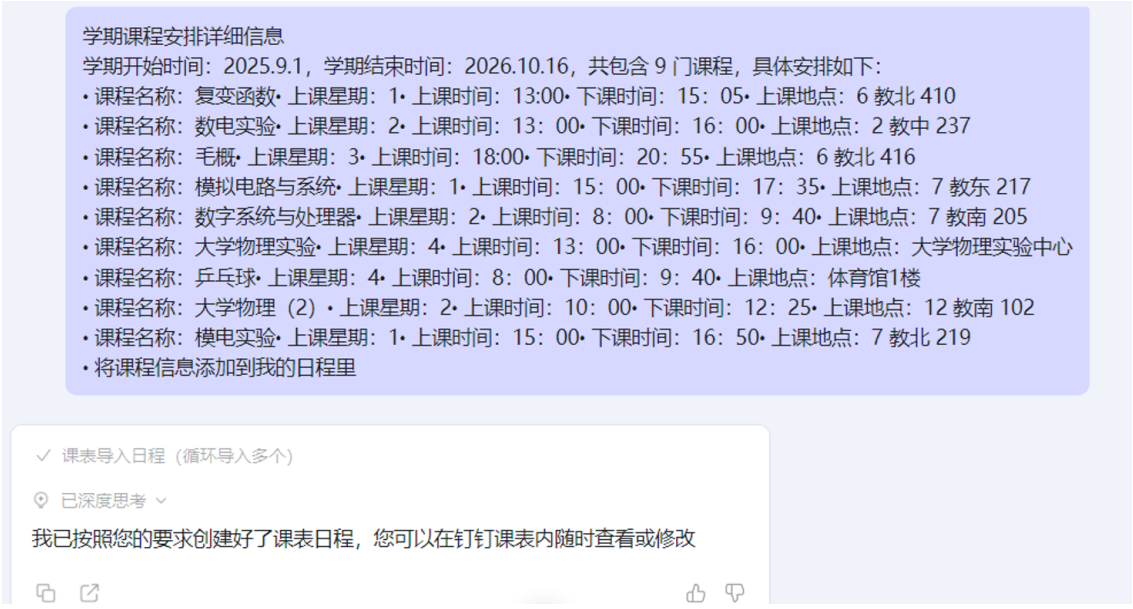
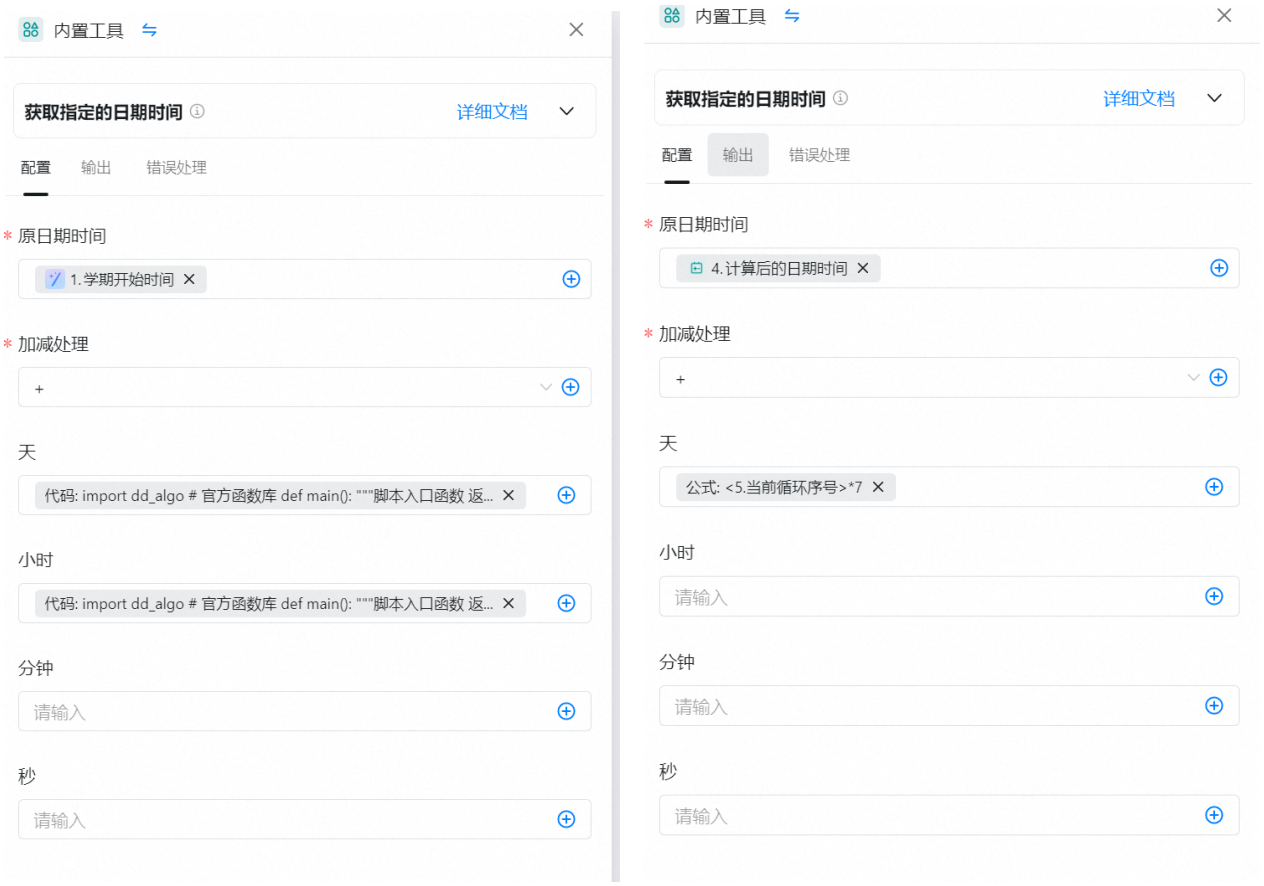


图 22: “课表导入日程”工作流的对话输入用户课表

图 23: “课表导入日程”工作流对于对话，成功的课表日程创建（以 11 月第一周为例）

通过这样的微观的公式代码嵌入，在工作流模块中建立了**高效稳定的变量控制，特别适用于重复操作的排班场景的智能体开发**；如在大企业的客服部门中，可以结合班次模板与个人请假信息，计算每名客服员工的首班起点与轮转周期，自动生成排班表，并创建钉钉日程提醒员工进行打卡、上岗、下班等，类似场景数不胜数，此开发思维将会是智能体开发的基本要素之一。



(a) 逐课程“第一次上课时间”的代码计算日程时刻的模块　　**(b)** 逐周逐课程“上课时间”的公式计算日程时刻的模块

图 24: “课表导入日程”工作流典型模块中的从“微观”到“微型”的公式代码嵌入

四　总结与展望

4.1　总结

本团队依托钉钉智能体平台，设计开发了“日程规划精灵”智能助手。深入分析现有平台日程排布存在的**逻辑错误、不支持连续安排**等痛点，以及学生等广大群体的日程管理需求后，针对性打造“空闲时间推荐器”“课表导入日程”“创建新日程”等原创工作流。

开发过程中，团队提炼出**“AI 决策 + 代码规范化变量 + 模块执行”融合开发方式**与**“宏观循环与微观公式代码多维嵌入”开发通则**。经实践验证，这些方法兼具可靠性与实用性，还能为其他场景的智能体开发提供普适参考，为行业同行积累了可借鉴的经验。

4.2　展望

(1) 应用场景：从细分到全域的延伸

“日程规划精灵”未来将为各领域日程管理减负：通过智能规划，缓解人们行程安排的压力，破除规划时的模糊与迷茫感。如图 25 所示，本团队试运行的“考研助手”网站，可接入“日程规划精灵”——借助智能助手分析每日学习量并给出计划推荐，助力考研群体

高效备考。而这只是场景拓展的缩影，未来助手还可接入**企业员工排程**等更多场景，持续拓宽应用边界。

图 25: “考研助手”试运行网站（可嵌入“日程规划精灵”辅助决策）

(2) 开发思想：从项目到行业的赋能

团队提出的两类开发思想，是针对当前信息产业“更灵活自然语言交互”等高层次需求的解决方案：一方面，“代码与大模型结合”“多维嵌入”的思想，能支撑国社科项目中“ERP 智能决策”等实际场景的智能体开发，推动其从社科研究转化为专利、教材、论文等科研成果，并落地到企业经营决策中，打通“产、学、研”一体化路径；另一方面，这类思想具备普适性，不仅局限于本项目，在所有涉及**自然语言交互**、**复杂指令控制**的场景中均可复用。

随着思想传播，会有更多非计算机专业的需求者参与智能体开发，让大模型应用突破“专业从业者”的局限，真正走进各行各业、千家万户。本团队在实践中实现的助手功能创新与开发思想提炼，也将为新兴的智能体开发领域提供有力支撑，助力行业持续进阶。